UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/684,662 | 10/14/2003 | Douglas R. Armstrong | 20002/15251 | 3392 |

34431 7590 07/25/2007

HANLEY, FLIGHT & ZIMMERMAN, LLC
150 S. WACKER DRIVE
SUITE 2100
CHICAGO, IL 60606

| EXAMINER | |
|---|---|
| WANG, BEN C | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2192 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 07/25/2007 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/684,662 | ARMSTRONG ET AL. | |
| | Examiner | Art Unit | |
| | Ben C. Wang | 2192 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *08 May 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-19 and 21-42* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19 and 21-42* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

1. Applicant's amendment dated May 8, 2007, responding to the Office action mailed Feb. 8, 2007 provided in the rejection of claims 1-41, wherein claims 1-19 and 21-41 are remained as original, claim 20 is canceled, and claim 42 is new.

Claims 1-19 and 21-42 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection – see *Hall, Hollingsworth-2, and Broberg* arts made of record, as applied hereto.

Claims 1-19 and 21-42 are pending in this application and presented for examination

### *Claim Rejections – 35 USC § 103(a)*

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made

2. Claims 1-2, 11-12, 21-22, 32-33, and 42 are rejected under 35 U.S.C. 103(a) as being unpatentable over R. J. Hall (*CPPROFJ: Aspect-capable Call Path Profiling of Multi-threaded Java Applications, 2002, IEEE, pp. 1-10*) (hereinafter 'Hall' - art made of record) in view of J. K. Hollingsworth (*Critical Path Profiling of Message Passing and*

*Shared-Memory Program, Jan. 1998, IEEE, pp. 1029-1040*) (hereinafter 'Hollingsworth-2' - art made of record)

3. **As to claim 1** (Original), Hall discloses a method of profiling a threaded program during program runtime, the method comprising: monitoring information exchanged between a processing unit and first and second threads executed by the processing unit (e.g., Abstract, Lines 10-21 – first, it provides two different call path oriented views on program performance, a server view and a thread view; the former helps one optimize for throughput, while the latter is useful for optimizing thread latency; the views incorporate a typed time notation for representing different program activities, such as monitor wait and thread preemption times; second, the new framework allows aspect-oriented program profiling, even when the original program was not designed in an aspect oriented fashion; finally, the approach is implemented in a too, CPPROFJ, an aspect-capable call path profile for Java™); determining, based on the information exchanged between the processing unit and the first and second threads, a wait time during which the first thread awaits a synchronization event (e.g., Sec. 1 – Introduction, 4$^{th}$ Par. – a third source of performance bottlenecks in modern applications is thread contention and other inter-thread dependencies; for example, an input/output processing thread may b stalled waiting for a thread marked with higher priority, event thought its data has arrived from the input device; it can often be better to mark the I/O processor as higher priority so that it can start the next read or write before letting the more compute-bound threads continue); and determining whether the wait time affects

the critical path of thread execution (e.g., Sec. 1 – Introduction, 5th Par, 3rd bullet – the thread view profiles report (a) time spent when actually running, (b) time spent sharing the CPU with equal priority threads, (c) time spent waiting while higher priority threads are running, (d) time spent waiting for a monitor, and (e) idle time waiting for other types of events).

Hall does not explicitly disclose determining, based on the information exchanged between the processing unit and the first and second threads, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree.

However, in an analogous art of Critical Path Profiling of Message Passing and Shared-Memory Program, Hollingsworth-2 discloses determining, based on the information exchanged between the processing unit and the first and second threads, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree (e.g., Fig. 1 – a program activity graph and calculating it critical path; Sec. 2 – Critical Path; Sec. 2.1 – Formal Definition of Critical Path).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hollingsworth-2 into the Hall's system to further provide determining, based on the information exchanged between the processing unit and the first and second threads, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree in Hall system.

The motivation is that it would further enhance the Hall's system by taking, advancing and/or incorporating Hollingsworth-2's system which offers significant

advantages of critical path profiling compared to metrics that simply add values for individual processes is that is provides a "global view" of the performance of a parallel computation that captures performance implications of the interactions between processes; and, in a distributed system, extracting a global view during the computation requires exchanging information between processes as once suggested by Hollingsworth-2 (e.g., Sec. 2 – Critical Path, 2nd Par.).

4. **As to claim 11** (Original), Hall discloses an article of manufacture comprising a machine-accessible medium having a plurality of machine-accessible instructions that, when executed, causes a machine to: monitor information exchanged between a processing unit and first and second threads executed by the processing unit (e.g., Abstract, Lines 10-21 – first, it provides two different call path oriented views on program performance, a server view and a thread view; the former helps one optimize for throughput, while the latter is useful for optimizing thread latency; the views incorporate a typed time notation for representing different program activities, such as monitor wait and thread preemption times; second, the new framework allows aspect-oriented program profiling, even when the original program was not designed in an aspect oriented fashion; finally, the approach is implemented in a too, CPPROFJ, an aspect-capable call path profile for Java™); determine, based on the information exchanged between the processing unit and the first and second threads, a wait time during which the first thread awaits a synchronization event (e.g., Sec. 1 – Introduction, 4th Par. – a third source of performance bottlenecks in modern applications is thread

contention and other inter-thread dependencies; for example, an input/output processing thread may b stalled waiting for a thread marked with higher priority, event thought its data has arrived from the input device; it can often be better to mark the I/O processor as higher priority so that it can start the next read or write before letting the more compute-bound threads continue); and determine whether the wait time affects the critical path of thread execution (e.g., Sec. 1 – Introduction, 5th Par, 3rd bullet – the thread view profiles report (a) time spent when actually running, (b) time spent sharing the CPU with equal priority threads, (c) time spent waiting while higher priority threads are running, (d) time spent waiting for a monitor, and (e) idle time waiting for other types of events).

Hall does not explicitly disclose determining, based on the information exchanged between the processing unit and the first and second threads, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree.

However, in an analogous art of Critical Path Profiling of Message Passing and Shared-Memory Program, Hollingsworth-2 discloses determining, based on the information exchanged between the processing unit and the first and second threads, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree (e.g., Fig. 1 – a program activity graph and calculating it critical path; Sec. 2 – Critical Path; Sec. 2.1 – Formal Definition of Critical Path).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hollingsworth-2 into the Hall's

system to further provide determining, based on the information exchanged between the processing unit and the first and second threads, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree in Hall system.

The motivation is that it would further enhance the Hall's system by taking, advancing and/or incorporating Hollingsworth-2's system which offers significant advantages of critical path profiling compared to metrics that simply add values for individual processes is that is provides a "global view" of the performance of a parallel computation that captures performance implications of the interactions between processes; and, in a distributed system, extracting a global view during the computation requires exchanging information between processes as once suggested by Hollingsworth-2 (e.g., Sec. 2 – Critical Path, 2nd Par.).

5. **As to claim 21** (Original), Hall discloses a method of profiling a threaded program during program runtime, the method comprising: monitoring information exchanged between a processing unit and first and second threads executed by the processing unit (e.g., Abstract, Lines 10-21 – first, it provides two different call path oriented views on program performance, a server view and a thread view; the former helps one optimize for throughput, while the latter is useful for optimizing thread latency; the views incorporate a typed time notation for representing different program activities, such as monitor wait and thread preemption times; second, the new framework allows aspect-oriented program profiling, even when the original program was not designed in an aspect oriented fashion; finally, the approach is implemented in a too, CPPROFJ, an aspect-capable call path profile for Java™); determining, based on the cross-thread

event and the information exchanged between the processing unit and the first and second threads (e.g., Sec. 1 – Introduction, 4th Par. – a third source of performance bottlenecks in modern applications is thread contention and other inter-thread dependencies; for example, an input/output processing thread may b stalled waiting for a thread marked with higher priority, event thought its data has arrived from the input device; it can often be better to mark the I/O processor as higher priority so that it can start the next read or write before letting the more compute-bound threads continue), a wait time during which the first thread awaits a synchronization event; and determining whether the wait time affects the critical path of thread execution (e.g., Sec. 1 – Introduction, 5th Par, 3rd bullet – the thread view profiles report (a) time spent when actually running, (b) time spent sharing the CPU with equal priority threads, (c) time spent waiting while higher priority threads are running, (d) time spent waiting for a monitor, and (e) idle time waiting for other types of events).

Hall does not explicitly disclose determining, based on the cross-thread event, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree.

However, in an analogous art of Critical Path Profiling of Message Passing and Shared-Memory Program, Hollingsworth-2 discloses determining, based on the cross-thread event, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree (e.g., Fig. 1 – a program activity graph and calculating it critical path; Sec. 2 – Critical Path; Sec. 2.1 – Formal Definition of Critical Path).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hollingsworth-2 into the Hall's system to further provide determining, based on the cross-thread event, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree in Hall system.

The motivation is that it would further enhance the Hall's system by taking, advancing and/or incorporating Hollingsworth-2's system which offers significant advantages of critical path profiling compared to metrics that simply add values for individual processes is that is provides a "global view" of the performance of a parallel computation that captures performance implications of the interactions between processes; and, in a distributed system, extracting a global view during the computation requires exchanging information between processes as once suggested by Hollingsworth-2 (e.g., Sec. 2 – Critical Path, 2nd Par.).

6. **As to claim 32** (Original), Hall discloses an article of manufacture comprising a machine-accessible medium having a plurality of machine-accessible instructions, when executed, causes a machine to: monitor information exchanged between a processing unit and first and second threads executed by the processing unit (e.g., Abstract, Lines 10-21 – first, it provides two different call path oriented views on program performance, a server view and a thread view; the former helps one optimize for throughput, while the latter is useful for optimizing thread latency; the views incorporate a typed time notation for representing different program activities, such as monitor wait and thread

preemption times; second, the new framework allows aspect-oriented program profiling, even when the original program was not designed in an aspect oriented fashion; finally, the approach is implemented in a too, CPPROFJ, an aspect-capable call path profile for Java™); determine, based on the cross-thread event and the information exchanged between the processing unit and the first and second threads (e.g., Sec. 1 – Introduction, 4th Par. – a third source of performance bottlenecks in modern applications is thread contention and other inter-thread dependencies; for example, an input/output processing thread may b stalled waiting for a thread marked with higher priority, event thought its data has arrived from the input device; it can often be better to mark the I/O processor as higher priority so that it can start the next read or write before letting the more compute-bound threads continue), a wait time during which the first thread awaits a synchronization event; and determine whether the wait time affects the critical path of thread execution (e.g., Sec. 1 – Introduction, 5th Par, 3rd bullet – the thread view profiles report (a) time spent when actually running, (b) time spent sharing the CPU with equal priority threads, (c) time spent waiting while higher priority threads are running, (d) time spent waiting for a monitor, and (e) idle time waiting for other types of events).

Hall does not explicitly disclose determining, based on the cross-thread event, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree.

However, in an analogous art of Critical Path Profiling of Message Passing and Shared-Memory Program, Hollingsworth-2 discloses determining, based on the cross-thread event, a critical path of thread execution and maintaining the critical path of

thread execution in a critical path tree (e.g., Fig. 1 – a program activity graph and calculating it critical path; Sec. 2 – Critical Path; Sec. 2.1 – Formal Definition of Critical Path).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hollingsworth-2 into the Hall's system to further provide determining, based on the cross-thread event, a critical path of thread execution and maintaining the critical path of thread execution in a critical path tree in Hall system.

The motivation is that it would further enhance the Hall's system by taking, advancing and/or incorporating Hollingsworth-2's system which offers significant advantages of critical path profiling compared to metrics that simply add values for individual processes is that is provides a "global view" of the performance of a parallel computation that captures performance implications of the interactions between processes; and, in a distributed system, extracting a global view during the computation requires exchanging information between processes as once suggested by Hollingsworth-2 (e.g., Sec. 2 – Critical Path, 2nd Par.).

7. **As to claim 2** (Original) (incorporating the rejection in claim 1) and **claim 12** (Original) (incorporating the rejection in claim 11), Hall discloses indicating that the wait time is of a high priority if the wait time affects the critical path of thread execution and indicating that the wait time is of a low priority if the wait time does not affect the critical path of thread execution (e.g., Sec. 1 – Introduction, 5th Par, 3rd bullet – the thread view

profiles report (a) time spent when actually running, (b) time spent sharing the CPU with equal priority threads, (c) time spent waiting while higher priority threads are running, (d) time spent waiting for a monitor, and (e) idle time waiting for other types of events).

8. **As to claim 20** (Cancelled)

9. **As to claim 22** (Original) (incorporating the rejection in claim 21) and **claim 33** (Original) (incorporating the rejection in claim 32), Hall discloses indicating that the wait time is of a high priority if the wait time affects the critical path of thread execution and indicating that the wait time is of a low priority if the wait time does not affect the critical path of thread execution (e.g., Sec. 1 – Introduction, 5th Par, 3rd bullet – the thread view profiles report (a) time spent when actually running, (b) time spent sharing the CPU with equal priority threads, (c) time spent waiting while higher priority threads are running, (d) time spent waiting for a monitor, and (e) idle time waiting for other types of events).

10. **As to claim 42** (New) (incorporating the rejection in claim 1), Hollingsworth-2 discloses a method wherein the information includes one or more timestamps (e.g., Fig. 1 – A program activity graph and calculating its critical path; Sec. 2 – Critical Path, 1st Par. – a simple definition of the critical path of a program is the longest, time-weighted sequence of events from the start of the program to its termination).

11. Claims 4-5, 7-9, 14-15, 17-19, 25-26, 28-30, 35-36, and 38-40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hall in view of Hollingsworth-2, and further in view of Broberg et al., (*Performance Optimization Using Critical Path Analysis in Multithreaded Programs on Multiprocessors, 1999, Psilander Grafiska, Karlskrona, Sweden, pp. 1-12*) (hereinafter 'Broberg' - art made of record)

12. **As to claim 4** (Original) (incorporating the rejection in claim 1) and **claim 14** (Original) (incorporating the rejection in claim 12), Hall and Hollingsworth-2 do not explicitly disclose a leaf is added to the critical path tree when the synchronization event is a signal event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is added to the critical path tree when the synchronization event is a signal event (e.g., Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is added to the critical path tree when the synchronization event is a signal event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the

optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

13. **As to claim 5** (Original) (incorporating the rejection in claim 1) **and claim 15** (Original) (incorporating the rejection in claim 12), Hall and Hollingsworth-2 do not explicitly disclose a leaf is removed from the critical path tree when the synchronization event is a wait event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is removed from the critical path tree when the synchronization event is a wait event (e.g., Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is removed from the critical path tree when the synchronization event is a wait event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

14. **As to claim 7** (Original) (incorporating the rejection in claim 1) and **claim 17** (Original) (incorporating the rejection in claim 12), Hall and Hollingsworth-2 do not explicitly disclose a leaf is removed from the critical path tree when the synchronization event is a block event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is removed from the critical path tree when the synchronization event is a block event (e.g., Sec. 3.1 – Finding the critical path in the optimal case, 4th Par., 5th Par.; Sec. 3.3 – Finding the critical path with CPU constraints, 3rd Par., 4th Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is removed from the critical path tree when the synchronization event is a block event in Hall-Hollingsworth-2 system. The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

15. **As to claim 8** (Original) (incorporating the rejection in claim 1) and **claim 18** (Original) (incorporating the rejection in claim 12), Hall and Hollingsworth-2 do not

explicitly disclose a leaf is removed from the critical path tree when the synchronization event is a suspend event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is removed from the critical path tree when the synchronization event is a suspend event (e.g., Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is removed from the critical path tree when the synchronization event is a suspend event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

16. **As to claim 9** (Original) (incorporating the rejection in claim 1) and **claim 19** (Original) (incorporating the rejection in claim 12), Hall and Hollingsworth-2 do not explicitly disclose a leaf is added to the critical path tree when the synchronization event is a resume event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is added to the critical path tree when the synchronization event is a resume event (e.g., Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is added to the critical path tree when the synchronization event is a resume event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

17. **As to claim 25** (Original) (incorporating the rejection in claim 22) and **claim 35** (Original) (incorporating the rejection in claim 32), Hall and Hollingsworth-2 do not explicitly disclose a leaf is added to the critical path tree when the synchronization event is a signal event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is added to the critical path tree when the synchronization event is a signal event (e.g.,

Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is added to the critical path tree when the synchronization event is a signal event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

18. **As to claim 26** (Original) (incorporating the rejection in claim 22) and **claim 36** (Original) (incorporating the rejection in claim 32), Hall and Hollingsworth-2 do not explicitly disclose a leaf is removed from the critical path tree when the synchronization event is a wait event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is removed from the critical path tree when the synchronization event is a wait event (e.g., Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is removed from the critical path tree when the synchronization event is a wait event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

19. **As to claim 28** (Original) (incorporating the rejection in claim 22) and **claim 38** (Original) (incorporating the rejection in claim 32), Ho Hall and Hollingsworth-2 do not explicitly disclose a leaf is removed from the critical path tree when the synchronization event is a block event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is removed from the critical path tree when the synchronization event is a block event (e.g., Sec. 3.1 – Finding the critical path in the optimal case, 4th Par., 5th Par.; Sec. 3.3 – Finding the critical path with CPU constraints, 3rd Par., 4th Par).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-

Hollingsworth-2's system to further provide a leaf is removed from the critical path tree when the synchronization event is a block event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

20. **As to claim 29** (Original) (incorporating the rejection in claim 22) and **claim 39** (Original) (incorporating the rejection in claim 32), Hall and Hollingsworth-2 do not explicitly disclose a leaf is removed from the critical path tree when the synchronization event is a suspend event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is removed from the critical path tree when the synchronization event is a suspend event (e.g., Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is removed from the critical path tree when the synchronization event is a suspend event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant

advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

21. **As to claim 30** (Original) (incorporating the rejection in claim 22) **and claim 40** (Original) (incorporating the rejection in claim 32), Hall and Hollingsworth-2 do not explicitly disclose a leaf is added to the critical path tree when the synchronization event is a resume event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a leaf is added to the critical path tree when the synchronization event is a resume event (e.g., Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a leaf is added to the critical path tree when the synchronization event is a resume event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

22. Claims 3, 6, 10, 13, 16, 24, 27, 31, 34, 37, and 41 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hall in view of Hollingsworth-2, and further in view of Intel™ Technology Journal (*Hyper-Threading Technology, Feb. 2002, Intel™ Technology Journal, pp. 1-66*) (hereinafter 'Intel' - art made of record)

23. **As to claim 3** (Original) (incorporating the rejection in claim 1) and **claim 13** (Original) (incorporating the rejection in claim 12), Hall and Hollingsworth-2 do not explicitly disclose a leaf is added to the critical path tree when the synchronization event is a fork event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Intel discloses a leaf is added to the critical path tree when the synchronization event is a fork event (e.g., P. 39, Right-Col, 1st Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Intel into the Hall-Hollingsworth-2's system to further provide a leaf is added to the critical path tree when the synchronization event is a fork event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Intel's system which offers significant advantages that Intel's Hyper-Threading Technology delivers two logical processors that can execute different tasks simultaneously using shared hardware resources; and, speculative pre-computation, a technique that improves the latency of single-threaded

applications by utilizing idle multithreading hardware resources to perform long-range data pre-fetches as once suggested by Intel (e.g., P. 2, 2nd Par., 3rd Par.).

24. **As to claim 6** (Original) (incorporating the rejection in claim 1) and **claim 16** (Original) (incorporating the rejection in claim 12), Hall and Hollingsworth-2 do not explicitly disclose a leaf is added to the critical path tree when the synchronization event is an entry event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Intel discloses a leaf is added to the critical path tree when the synchronization event is an entry event (e.g., P. 38, Sec. of Multi-Entry Threading).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Intel into the Hall-Hollingsworth-2's system to further provide a leaf is added to the critical path tree when the synchronization event is an entry event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Intel's system which offers significant advantages that Intel's Hyper-Threading Technology delivers two logical processors that can execute different tasks simultaneously using shared hardware resources; and, speculative pre-computation, a technique that improves the latency of single-threaded applications by utilizing idle multithreading hardware resources to perform long-range data pre-fetches as once suggested by Intel (e.g., P. 2, 2nd Par., 3rd Par.).

25. **As to claim 10** (Original) (incorporating the rejection in claim 1), Hall and Hollingsworth-2 do not explicitly disclose comparing a number of active threads to a number of processing resources to determine a utilization factor.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Intel discloses comparing a number of active threads to a number of processing resources to determine a utilization factor (e.g., P. 2, 3rd Par.; P.3, 2nd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Intel into the Hall-Hollingsworth-2's system to further provide comparing a number of active threads to a number of processing resources to determine a utilization factor in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Intel's system which offers significant advantages that Intel's Hyper-Threading Technology delivers two logical processors that can execute different tasks simultaneously using shared hardware resources; and, speculative pre-computation, a technique that improves the latency of single-threaded applications by utilizing idle multithreading hardware resources to perform long-range data pre-fetches as once suggested by Intel (e.g., P. 2, 2nd Par., 3rd Par.).

26. **As to claim 24** (Original) (incorporating the rejection in claim 22) and **claim 34** (Original) (incorporating the rejection in claim 32), Hall and Hollingsworth-2 do not explicitly disclose a leaf is added to the critical path tree when the synchronization event is a fork event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Intel discloses a leaf is added to the critical path tree when the synchronization event is a fork event (e.g., P. 39, Right-Col, 1st Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Intel into the Hall-Hollingsworth-2's system to further provide a leaf is added to the critical path tree when the synchronization event is a fork event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Intel's system which offers significant advantages that Intel's Hyper-Threading Technology delivers two logical processors that can execute different tasks simultaneously using shared hardware resources; and, speculative pre-computation, a technique that improves the latency of single-threaded applications by utilizing idle multithreading hardware resources to perform long-range data pre-fetches as once suggested by Intel (e.g., P. 2, 2nd Par., 3rd Par.).

27. **As to claim 27** (Original) (incorporating the rejection in claim 22) and **claim 37** (Original) (incorporating the rejection in claim 32), Hall and Hollingsworth-2 do not

explicitly disclose a leaf is added to the critical path tree when the synchronization event is an entry event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Intel discloses a leaf is added to the critical path tree when the synchronization event is an entry event (e.g., P. 38, Sec. of Multi-Entry Threading).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Intel into the Hall-Hollingsworth-2's system to further provide a leaf is added to the critical path tree when the synchronization event is an entry event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Intel's system which offers significant advantages that Intel's Hyper-Threading Technology delivers two logical processors that can execute different tasks simultaneously using shared hardware resources; and, speculative pre-computation, a technique that improves the latency of single-threaded applications by utilizing idle multithreading hardware resources to perform long-range data pre-fetches as once suggested by Intel (e.g., P. 2, 2nd Par., 3rd Par.).

28. **As to claim 31** (Original) (incorporating the rejection in claim 22) and **claim 41** (Original) (incorporating the rejection in claim 32), Hall and Hollingsworth-2 do not explicitly disclose comparing a number of active threads to a number of processing resources to determine a utilization factor.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Intel discloses comparing a number of active threads to a number of processing resources to determine a utilization factor (e.g., P. 2, 3rd Par.; P.3, 2nd Par.).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Intel into the Hall-Hollingsworth-2's system to further provide comparing a number of active threads to a number of processing resources to determine a utilization factor in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Intel's system which offers significant advantages that Intel's Hyper-Threading Technology delivers two logical processors that can execute different tasks simultaneously using shared hardware resources; and, speculative pre-computation, a technique that improves the latency of single-threaded applications by utilizing idle multithreading hardware resources to perform long-range data pre-fetches as once suggested by Intel (e.g., P. 2, 2nd Par., 3rd Par.).

29. Claim 23 is rejected under 35 U.S.C. 103(a) as being unpatentable over Hall in view of Hollingsworth-2 and Broberg and further in view of Intel.

30. **As to claim 23** (Original) (incorporating the rejection in claim 22), Hall and Hollingsworth-2 do not explicitly disclose a method wherein the cross-thread event is

selected from a group consisting of a fork event, an entry event, a signal event, a wait event, a suspend event, a resume event, and a block event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Broberg discloses a method wherein the cross-thread event is selected from a group consisting of a signal event, a wait event, a suspend event, a resume event, and a block event (e.g., Sec. 2 – Overview of the Critical Path Analysis, 2nd Par., 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 3rd Par.; Sec. 3.1 – Finding the critical path in the optimal case, 4th Par., 5th Par.; Sec. 3.3 – Finding the critical path with CPU constraints, 3rd Par., 4th Par).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Broberg into the Hall-Hollingsworth-2's system to further provide a method wherein the cross-thread event is selected from a group consisting of a signal event, a wait event, a suspend event, a resume event, and a block event in Hall-Hollingsworth-2 system.

The motivation is that it would further enhance the Hall-Hollingsworth-2's system by taking, advancing and/or incorporating Broberg's system which offers significant advantages for finding the critical path of the multi-threaded program and the optimization is only done on those specific code segments of the program as once suggested by Broberg (e.g., Abstract).

Furthermore, Hall, Hollingsworth-2, and Broberg do not explicitly disclose a method wherein the cross-thread event is selected from a group consisting of a fork event, an entry event.

However, in an analogous art of Performance Optimization using Critical Path Analysis in Multithreaded Programs on Multiprocessors, Intel discloses a method wherein the cross-thread event is selected from a group consisting of a fork event, an entry event (e.g., P. 39, Right-Col, 1st Par.; P. 38, Sec. of Multi-Entry Threading).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Intel into the Hall-Hollingsworth-2- Broberg's system to further provide a method wherein the cross-thread event is selected from a group consisting of a fork event, an entry event in Hall-Hollingsworth-2- Broberg system.

The motivation is that it would further enhance the Hall-Hollingsworth-2-Broberg's system by taking, advancing and/or incorporating Intel's system which offers significant advantages that Intel's Hyper-Threading Technology delivers two logical processors that can execute different tasks simultaneously using shared hardware resources; and, speculative pre-computation, a technique that improves the latency of single-threaded applications by utilizing idle multithreading hardware resources to perform long-range data pre-fetches as once suggested by Intel (e.g., P. 2, 2nd Par., 3rd Par.).

***Conclusion***

31. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TUAN DAM
SUPERVISORY PATENT EXAMINER

BCW

July 20, 2007